

Re: ext3 allocate-with-reservation latencies

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-04/3034.html>

From: Mingming Cao (*cmm_at_us.ibm.com*)

Date: 04/11/05

To: "Stephen C. Tweedie" <sct@redhat.com>

Date: Mon, 11 Apr 2005 11:38:30 -0700

On Mon, 2005-04-11 at 12:48 +0100, Stephen C. Tweedie wrote:

> *Hi,*

>

> *On Fri, 2005-04-08 at 19:10, Mingming Cao wrote:*

>

>> *It still needs to be done under locking to prevent us from expanding
>>> over the next window, though. And having to take and drop a spinlock a
>>> dozen times or more just to find out that there are no usable free
>>> blocks in the current block group is still expensive, even if we're not
>>> actually fully unlinking the window each time.*

>

>> *Isn't this a rare case? The whole group is relatively full and the free
>> bits are all reserved by other files.*

>

> *Well, that's not much different from the case where there are usable
> bits but they are all near the end of the bitmap. And that's common
> enough as you fill up a block group with small files, for example. Once
> the bg becomes nearly full, new file opens-for-write will still try to
> allocate in that bg (because that's where the inode was allocated), but
> as it's a new fd we have no prior knowledge of where in the bh to
> allocate, and we'll have to walk it to the end to find any free space.
> This is the access pattern I'd expect of (for example) "tar xvjf
> linux-2.6.12.tar.bz2", not exactly a rare case.*

>

Okey.

>> *Probably we should avoid trying
>> to make reservation in this block group at the first place*

>

> *Not in this case, because it's the "home" of the file in question, and
> skipping to another bg would just leave useful space empty --- and that
> leads to fragmentation.*

>

I agree. We should not skip the home block group of the file. I guess what I was suggesting is, if allocation from the home group failed and we continuing the linear search the rest of block groups, we could

Linux-Kernel: Re: ext3 allocate-with-reservation latencies

probably try to skip the block groups without enough usable free blocks to make a reservation. Checking for the number of free blocks left in the quarry bg is a good way, but probably not good enough, since those free blocks might already being reserved.

- > > *You are proposing that we hold the read lock first, do the window search*
- > > *and bitmap scan, then once we confirm there is free bit in the candidate*
- > > *window, we grab the write lock and update the tree?*
- >
- > *No, I'm suggesting that if we need the write lock for tree updates, we*
- > *may still be able to get away with just a read lock when updating an*
- > *individual window. If all we're doing is winding the window forwards a*
- > *bit, that's not actually changing the structure of the tree.*
- >
- > > *However I am still worried that the rw lock will allow concurrent files*
- > > *trying to lock the same window at the same time.*
- >
- > *Adding a new window will need the write lock, always. But with the read*
- > *lock, we can still check the neighbouring windows (the structure is*
- > *pinned so those remain valid), so advancing a window with just that lock*
- > *can still be done without risking overlapping the next window.*
- >

Ah.. I see the win with the read lock now: once the a reservation window is added, updating it (either winding it forward and or searching for a available window) probably is the majority of the operations on the tree, and using read lock for that should help reduce the latency.

I will work on a patch for Lee to try sometime tonight.

Thanks,

Mingming

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@vger.kernel.org
More majordomo info at <http://vger.kernel.org/majordomo-info.html>
Please read the FAQ at <http://www.tux.org/lkml/>